Regressor Basis Learning for Anchored Super-Resolution

Eirikur Agustsson
Computer Vision Lab
D-ITET, ETH Zurich
aeirikur@vision.ee.ethz.ch

Radu Timofte
Computer Vision Lab
D-ITET, ETH Zurich
radu.timofte@vision.ee.ethz.ch

Luc Van Gool
ESAT, KU Leuven
D-ITET, ETH Zurich
vangool@vision.ee.ethz.ch

***Abstract*—A+ aka Adjusted Anchored Neighborhood Regression - is a state-of-the-art method for exemplar-based single image super-resolution with low time complexity at both train and test time. By robustly training a clustered regression model over a low-resolution dictionary, its performance keeps improving with the dictionary size - even when using tens of thousands of regressors. However, this can pose a memory issue where the model size can grow to more than a gigabyte, limiting applicability in memory constrained scenarios. To address this, we propose Regressor Basis Learning (RB), a novel variant of A+ where we restrict the regressor set to a learned low-dimensional subspace, such that each regressor is coded as a linear combination of few basis regressors. We learn the regressor basis by alternating between closed form solutions of the optimal coding of the regressor set (given the basis) and the optimal regressor basis (given the coding). We validate RB on several standard benchmarks and achieve comparable performance to A+ but by using orders of magnitude fewer basis regressors, *ie.* 32 basis regressors instead of 1024 regressors. This makes our RB method ideal for memory constrained applications.**

I. Introduction

Image resizing is an ubiquitous image operation on nowadays for displays and editing software. While downsampling or downsizing does not pose problems, since part of the information is dropped, increasing the size or super-resolving the images is challenging and an open problem. Image super-resolution is an ill posed problem, since for a low-resolution (LR) image patch there can be a multitude of high-resolution (HR) corresponding patches, causing an inherent ambiguity in choosing the right HR patch.

Single image super-resolution (SR) research spans decades. The interpolation methods such as nearest neighbor, bilinear or bicubic were among the first to produce low time and memory complexity results and are still in broad use. While fast and simple, the interpolation methods are not able to restore missing high frequencies and produce artifacts such as edge halos and blur. Other directions try to embed knowledge and different image priors at level of edges and natural image statistics. The example based SR direction [9] is the one currently most active and usually the methods are more involved in comparison with fast interpolations. According to the source of information used in the SR process, the example based methods can be roughly categorized into internal dictionary methods that use solely the information extracted from the input LR image and external dictionary methods with models and priors extracted from extra exemplars of LR and HR natural images.

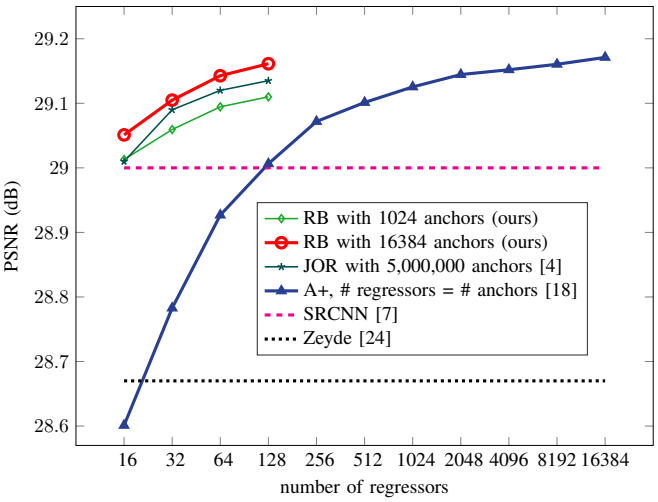


Fig. 1. Average PSNR [dB] on (Set14, $\times 3$) vs. number of regressors and number of anchor points for our RB method, A+ [18] and JOR [4]. Our RB uses much fewer regressors ($> 32\times$) than A+ and much fewer anchors than JOR ($> 305\times$) rendering it a memory efficient method. Details in Section IV.

Internal dictionary SR methods [10], [11] usually have high complexities since they need to extract and organize the data on-line from the input LR image. Glasner *et al.* [10] gradually increase the image resolution by using the patch redundancy / self-similarity from the natural images. Recently, Huang *et al.* [11] follow the same direction and use transformed self-exemplars for improved performance.

External dictionary SR methods [17], [18], [20], [6], [3], [15], [4], [21], [9], [22], [1], [8], [14], on the other hand, by using external data and moving offline most of the computation required to extract and build useful priors and models, are able to reach better on-line complexities and SR performance. Among them, we can distinguish various assumptions and principles that make these methods successful. Chang *et al.* [2] assume local manifolds for both LR and corresponding HR image patches, proposing a neighbor embedding method. Yang *et al.* [23] assume sparsity and perform sparse coding over learned dictionaries of LR and HR patches in the ScSR method. By doing so they need considerably smaller dictio-

naries than Chang *et al.* but the time complexity increases as the computation of a sparse coding is required. Zeyde *et al.* [24] speed up the overall framework and uses K-SVD and orthogonal matching pursuit to enforce sparsity. Timofte *et al.* [17] go further with Anchored Neighborhood Regression (ANR) and compute, over the same dictionary, offline anchored linear regressors from LR to HR, transforming the SR task into a linear search followed by a regression of the LR input patch. The follow-up Adjusted ANR (A+) method [18] computes the regressors from training LR and HR patches instead of the small dictionary and thus uses better the prior data for improved performance. Dai *et al.* [4] jointly optimize regressors (JOR) over the training LR and HR patches and achieve comparable performance with fewer regressors, at the expense of increased complexity in the selection of regressors. Schulter *et al.* [15] map LR to HR by random forests and ridge regressors as in A+. Dong *et al.* [6] apply a convolutional neural network (SRCNN) for end-to-end LR to HR mapping, while Wang *et al.* [21] in their cascade of sparse coding network (CSCN) combine ideas from convolutional neural networks [7], [12] and ScSR[23].

A number of techniques can be applied to any super-resolution method to achieve improved results and seven are demonstrated in [20]. Other works study the semantic super-resolution [19] and how useful super-solution is for other vision tasks [5].

In this paper, we propose to learn a regressor basis (RB) for anchored super-resolution within the framework of A+ [18] to greatly reduce the memory footprint of the method. Our RB method keeps anchor points and the general pipeline of A+ while reducing the regressors to codings over a compact learned regressor basis. Thus, with a fraction of the number of regressors of A+, RB achieves comparable performance (see Fig. 1). This is a remarkable result especially for memory constrained applications and devices and allows for much larger numbers of anchors within the A+ framework at order(s) of magnitude lower memory requirements. At the same time, as shown in our experiments, RB has a lower (memory and time) complexity than JOR requiring orders of magnitude fewer stored anchor points than JOR for equal number of regressors and PSNR performance.

The remainder of the paper is structured as follows. In Section II we briefly reintroduce A+ [18] with a slightly modified notation, which makes the formulation of our Regressor Basis Learning in Section III more succinct. In Section IV we describe the experimental setup and discuss the results, to then conclude the paper in Section V.

II. Anchored Regression (A+)

Adjusted Anchored Neighborhood Regression (A+) [18] robustly trains a family of regressors corresponding to a number of anchor points (or atoms) from a low-resolution dictionary, which is learned from the training data with K-SVD as in [24], [17]. The training data consists of extracted low resolution patches $\mathcal{X} \subset \mathbb{R}^{d_l}$ and corresponding high resolution patches $\mathcal{Y} \subset \mathbb{R}^{d_h}$. We denote the low resolution dictionary as $\mathbf{D}^l = [\mathbf{d}_1^l, \cdots, \mathbf{d}_N^l] \in \mathbb{R}^{d_l \times N}$ and the corresponding high resolution dictionary as $\mathbf{D}^h = [\mathbf{d}_1^h, \cdots, \mathbf{d}_N^h] \in \mathbb{R}^{d_h \times N}$, where N is the number of atoms/anchor points. For each dictionary atom $\mathbf{d}_i^l$, A+ obtains the K nearest samples $\mathbf{X}_i = [\mathbf{x}_1^i, \cdots, \mathbf{x}_K^i] \in \mathbb{R}^{d_l \times K}$ from $\mathcal{X}$. From the corresponding high resolution patches $\mathbf{Y}_i = [\mathbf{y}_1^i, \cdots, \mathbf{y}_K^i] \in \mathbb{R}^{d_h \times K}$, A+ uses (vector) ridge regression to learn a regressor

$$\mathbf{W}_i = \underset{\mathbf{W}' \in \mathbb{R}^{d_h \times d_l}}{\arg\min} \sum_{k=1}^{K} \|\mathbf{W}\mathbf{x}_k^i - \mathbf{y}_k^i\|_2^2 + \lambda \|\mathbf{W}\|_F^2 \quad (1)$$

$$= \underset{\mathbf{W}' \in \mathbb{R}^{d_h \times d_l}}{\arg\min} \|\mathbf{W}\mathbf{X}_i - \mathbf{Y}_i\|_F^2 + \lambda \|\mathbf{W}\|_F^2, \quad (2)$$

where $\|\cdot\|_F$ is the Frobenius norm.

The regressor $\mathbf{W}_i$ can be obtained through the closed form solution:

$$\mathbf{W}_i = \mathbf{Y}_i \mathbf{X}_i^T (\mathbf{X}_i \mathbf{X}_i^T + \lambda \mathbf{I}_{d_l})^{-1}, \quad (3)$$

where $\mathbf{I}_{d_l}$ denotes the $d_l \times d_l$ identity matrix.

At test time, for a sample $\mathbf{x} \in \mathbb{R}^{d_l}$ the nearest anchor $\mathbf{d}_{\gamma(\mathbf{x})}^l$ is found (with index $\gamma(\mathbf{x})$) and the corresponding regressor is applied:

$$\hat{\mathbf{y}} = \mathbf{W}_{\gamma(\mathbf{x})} \mathbf{x}. \quad (4)$$

In contrast to clustered regression, *ie.* where the data is clustered and a model trained on each cluster, the fixed neighbourhood size K ensures that each regressor will be robustly trained. Since each regressor $\mathbf{W}_i$ is trained independently and requires only solving a linear problem over $d_h \times d_l$ variables, the training is also very efficient.

Thus, A+ has been successfully applied using as much as 65,536 regressors [20], [16], achieving state-of-the-art results. However, for N regressors (and atoms), the memory complexity is $O(Nd_l d_h + Nd_l)$, *ie.* almost 1.2GB for x4 upscaling ($N = 65536, d_l = 31, d_h = 144$, single precision).

This motivates our proposed method, detailed in the next section.

III. Regressor Basis Learning

In this section, we describe the proposed method, Regressor Basis Learning (RB) for anchored super resolution. We remain inside the framework of A+, using the same dictionaries $\mathbf{D}^l, \mathbf{D}^h$ and N neighbourhoods $\mathbf{X}_i, \mathbf{Y}_i$ for training. However, we restrict the regressors $\mathbf{W}_i$ to a R-dimensional linear subspace of $\mathbb{R}^{d_h \times d_l}$, *ie.* we learn a basis $\tilde{\mathbf{W}}^{(1)}, \cdots, \tilde{\mathbf{W}}^{(R)}$ such that each regressor is expressed as a linear combination

$$\mathbf{W}_i = \sum_{j=1}^{R} \alpha_i^{(j)} \tilde{\mathbf{W}}^{(j)}.$$

Using a small number of basis regressors (br.), $R \ll N$, we then only need to store the basis, $O(Rd_l d_h)$, and the coding of each regressor, $O(NR)$, compared to $O(Nd_l d_h)$ required by N regressors of A+.

The question remains on how to obtain a good basis $\tilde{\mathbf{W}} = [\tilde{\mathbf{W}}^{(1)}, \cdots, \tilde{\mathbf{W}}^{(R)}] \in \mathbb{R}^{d_h \times Rd_l}$ and a coding $\boldsymbol{\alpha}_i = [\alpha_i^{(1)}, \cdots, \alpha_i^{(R)}]^T \in \mathbb{R}^R$ for each $\mathbf{W}_i$.

The trivial approach would be to apply a PCA-approximation to $\mathbf{W}_1, \cdots, \mathbf{W}_N$, but our experiments show that is far from the optimal strategy (see Figure 2).

Instead, we consider the training objective of A+ (*ie.* equation (2)), re-parametrized over $\tilde{\mathbf{W}}$ and $\boldsymbol{\alpha}_i$, taken jointly over all neighbourhoods:

$$L := \sum_{i=1}^{N} \left\| \left(\sum_{j=1}^{R} \alpha_i^{(j)} \tilde{\mathbf{W}}^{(j)} \right) \mathbf{X}_i - \mathbf{Y}_i \right\|_F^2 + \lambda \|\tilde{\mathbf{W}}\|_F^2, \quad (5)$$

which we want to minimize over all $\tilde{\mathbf{W}}^{(1)}, \cdots, \tilde{\mathbf{W}}^{(R)}$ and $\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_N$. While the objective in (5) seems unmanageable to optimize over thousands of neighbourhoods N and so many parameters, we will see that by fixing either $\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_N$ or $\tilde{\mathbf{W}}^{(1)}, \cdots, \tilde{\mathbf{W}}^{(R)}$, the problem turns into a ridge regression for the other set of parameters, enabling us perform alternating optimization over the parameter sets.

We will start by looking at the simpler case, when $\tilde{\mathbf{W}} = [\tilde{\mathbf{W}}^{(1)}, \cdots, \tilde{\mathbf{W}}^{(R)}]$ is fixed. In this case, we can separately minimize each term

$$L_i := \left\| \left(\sum_{j=1}^{R} \alpha_i^{(j)} \tilde{\mathbf{W}}^{(j)} \right) \mathbf{X}_i - \mathbf{Y}_i \right\|_F^2 \quad (6)$$

of (5) over $\boldsymbol{\alpha}_i$. We can now compute

$$L_i = \left\| \sum_{j=1}^{R} \alpha_i^{(j)} \left(\tilde{\mathbf{W}}^{(j)} \mathbf{X}_i \right) - \mathbf{Y}_i \right\|_F^2 \quad (7)$$

$$= \left\| \sum_{j=1}^{R} \alpha_i^{(j)} \operatorname{vec}(\tilde{\mathbf{W}}^{(j)} \mathbf{X}_i) - \operatorname{vec}(\mathbf{Y}_i) \right\|_2^2 \quad (8)$$

$$= \left\| [\operatorname{vec}(\tilde{\mathbf{W}}^{(1)} \mathbf{X}_i), \cdots, \operatorname{vec}(\tilde{\mathbf{W}}^{(R)} \mathbf{X}_i)] \boldsymbol{\alpha}_i - \operatorname{vec}(\mathbf{Y}_i) \right\|_2^2, \quad (9)$$

which gives the optimal

$$\boldsymbol{\alpha}_i = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \operatorname{vec}(\mathbf{Y}_i), \quad (10)$$

with $\mathbf{A} := [\operatorname{vec}(\tilde{\mathbf{W}}^{(1)} \mathbf{X}_i), \cdots, \operatorname{vec}(\tilde{\mathbf{W}}^{(R)} \mathbf{X}_i)] \in \mathbb{R}^{d_h K \times R}$.

Now suppose $\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_N$ are fixed. Since $\tilde{\mathbf{W}} = [\tilde{\mathbf{W}}^{(1)}, \cdots, \tilde{\mathbf{W}}^{(R)}]$ is now shared among all N neighbourhoods, we cannot optimized each L_i separately. Instead, we will formulate (5) as a ridge regression problem over all $\mathbf{X}_i, \mathbf{Y}_i$ and $\boldsymbol{\alpha}_i$. To this end, we define the matrix

$$\tilde{\boldsymbol{\alpha}}_i := \begin{bmatrix} \alpha_i^{(1)} \mathbf{I}_{d_l} \\ \vdots \\ \alpha_i^{(R)} \mathbf{I}_{d_l} \end{bmatrix} \in \mathbb{R}^{(Rd_l) \times d_l}, \quad (11)$$

such that we can write

$$\sum_{j=1}^{R} \alpha_i^{(j)} \tilde{\mathbf{W}}^{(j)} = \tilde{\mathbf{W}} \tilde{\boldsymbol{\alpha}}_i. \quad (12)$$

We can now put (5) into matrix form

$$L = \sum_{i=1}^{N} \left\| \tilde{\mathbf{W}} \tilde{\boldsymbol{\alpha}}_i \mathbf{X}_i - \mathbf{Y}_i \right\|_F^2 + \lambda \|\tilde{\mathbf{W}}\|_F^2 \quad (13)$$

$$= \left\| \tilde{\mathbf{W}} [\tilde{\boldsymbol{\alpha}}_1 \mathbf{X}_1, \ldots, \tilde{\boldsymbol{\alpha}_N} \mathbf{X}_N] - [\mathbf{Y}_1, \ldots, \mathbf{Y}_N] \right\|_F^2 + \lambda \|\tilde{\mathbf{W}}\|_F^2. \quad (14)$$

Denoting $\hat{\mathbf{Y}} := [\mathbf{Y}_1, \ldots, \mathbf{Y}_N] \in \mathbb{R}^{d_h \times NK}$ and $\hat{\mathbf{X}} := [\tilde{\boldsymbol{\alpha}}_1 \mathbf{X}_1, \ldots, \tilde{\boldsymbol{\alpha}_N} \mathbf{X}_N] \in \mathbb{R}^{Rd_l \times NK}$, we can view (14) as a ridge regression problem with $\hat{\mathbf{X}}$ and $\hat{\mathbf{Y}}$ as the observations. This gives us a closed form solution

$$\tilde{\mathbf{W}} = \hat{\mathbf{Y}} \hat{\mathbf{X}}^T (\hat{\mathbf{X}} \hat{\mathbf{X}}^T + \lambda \mathbf{I}_{Rd_l})^{-1} \quad (15)$$

that minimizes (5). While the involved matrices, $\hat{\mathbf{Y}}$ and $\hat{\mathbf{X}}$, can contain millions of entries (for $N > 1000, K > 1000$), we can efficiently compute:

$$\hat{\mathbf{X}} \hat{\mathbf{X}}^T = \sum_{i=1}^{N} (\tilde{\boldsymbol{\alpha}}_i \mathbf{X}_i)(\tilde{\boldsymbol{\alpha}}_i \mathbf{X}_i)^T = \sum_{i=1}^{N} \tilde{\boldsymbol{\alpha}}_i (\mathbf{X}_i \mathbf{X}_i^T) \tilde{\boldsymbol{\alpha}}_i{}^T =: \mathbf{Z} \quad (16)$$

and

$$\hat{\mathbf{Y}} \hat{\mathbf{X}}^T = \sum_{i=1}^{N} \mathbf{Y}_i (\tilde{\boldsymbol{\alpha}}_i \mathbf{X}_i)^T = \sum_{i=1}^{N} (\mathbf{Y}_i \mathbf{X}_i^T) \tilde{\boldsymbol{\alpha}}_i{}^T =: \mathbf{Q}, \quad (17)$$

such that computing $\tilde{\mathbf{W}} = \mathbf{Q}(\mathbf{Z} + \lambda \mathbf{I}_{Rd_l})^{-1}$ only involves the matrices $\mathbf{Q}$ and $\mathbf{Z}$ of size $d_h \times Rd_l$ and $Rd_l \times Rd_l$, respectively.

Alternating between the closed form solutions (10) and (15) gives us our Regressor Basis Learning algorithm, detailed in Algorithm 1. Since the global loss L in equation (5) is reduced in each iteration (and bounded by 0 below), convergence is guaranteed. In practice we only need a couple iterations to converge to a good solution.

Algorithm 1 Regressor Basis Learning

Input: initial $\tilde{\mathbf{W}} = [\tilde{\mathbf{W}}^{(1)} \ldots \tilde{\mathbf{W}}^{(R)}]$, $(\mathbf{X}_i)_{i=1}^N, (\mathbf{Y}_i)_{i=1}^N$
Output: final $\tilde{\mathbf{W}}$, codings $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_N$
while Not Converged **do**
  for $i = 1 \ldots N$ **do**
    $\mathbf{A} \leftarrow [\operatorname{vec}(\tilde{\mathbf{W}}^{(1)} \mathbf{X}_i), \cdots, \operatorname{vec}(\tilde{\mathbf{W}}^{(R)} \mathbf{X}_i)]$
    $\boldsymbol{\alpha}_i \leftarrow (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \operatorname{vec}(\mathbf{Y}_i)$
  end for
  $\mathbf{Z} \leftarrow \mathbf{0}$
  $\mathbf{Q} \leftarrow \mathbf{0}$
  for $i = 1 \ldots N$ **do**
    $\mathbf{Z} \leftarrow \mathbf{Z} + \tilde{\boldsymbol{\alpha}}_i (\mathbf{X}_i \mathbf{X}_i^T) \tilde{\boldsymbol{\alpha}}_i^T$
    $\mathbf{Q} \leftarrow \mathbf{Q} + (\mathbf{Y}_i \mathbf{X}_i^T) \tilde{\boldsymbol{\alpha}}_i^T$
  end for
  $\tilde{\mathbf{W}} \leftarrow \mathbf{Q}(\mathbf{Z} + \lambda \mathbf{I}_{Rd_l})^{-1}$
end while

IV. Experiments

In this section we study the relationship between the performance and the internal parameters of our method, the number of basis regressors and anchor points used. Furthermore, we report the performance of our proposed RB method and compare with other state-of-the-art methods.

A. Datasets and methods

We adhere to the same benchmark as in [17], [18]. That is, training on 91 images [23], testing on another 3 datasets (Set5, Set14, B100), and using bicubic downscaling for obtaining LR images.

Set5 and **Set14** consist of 5 and 14 images, respectively, and are used for validation of single-image super-resolution methods. We use Set14 and upscaling factor $\times 3$ to study the internal parameters of our method and for comparing the performance to other methods for different number of regressors and anchor points.

B100 consists of the 100 testing images of the Berkeley Segmentation Dataset (BSDS300)[13], a widely used dataset of natural images originally designed for image segmentation.

Compared methods We compare our RB method with other state-of-the-art methods, in particular: NE+LLE (Neighbour Embedding + Locally Linear Embedding, similar to Chang *et al.* [2], codes by Timofte *et al.* [18]), the efficient sparse coding method of Zeyde *et al.* [24], ANR (Anchored Neghborhood Regression) as well as A+ (Adjusted Anchored Neighborhood Regression) by Timofte *et al.* [17], [18], SRCNN (Convolutional Neural Network) of Dong *et al.* [7], and JOR (Jointly Optimized Regressors) of Dai *et al.* [4]. All methods already briefly described in the introductory sections.

B. Implementation details

We implement our Regressor Basis Learning (RB) method by extending the codes of A+ [18] and using the same validation benchmark, which also ensures a fair comparison.

We extract 5 million low and high resolution patches as in A+ [18] and use the same trained dictionary across methods for a given number of anchor points. Also we fix the size of the neighborhood of training patches used for computing each anchored regressor to 2048 as in A+. We initialize our RB algorithm with a PCA approximation of the regressors obtained with A+ and perform 4 iterations for the basis learning (Algorithm 1). The results are very stable with respect to the regularization parameter λ, which we fix as $10^{-5}N$ (where N is the number of anchors).

In other aspects, we also stay within the A+ framework[18], using the same codes for patch/feature extraction and benchmarking.

C. Parameters versus performance

Our RB shares with A+ a number of parameters: number of training samples, neighborhood size, and number of anchors (or atoms in dictionary). The number of iterations for learning the basis and the number of regressors forming the basis are specific to RB. We fix the number of training samples (5 million) and the neighborhood size (2048) as in A+ and refer to [18] for a discussion of them. As shown in [18], [20] using more training samples and/or more anchors/regressors usually leads to improved performance for the anchored regression methods such as A+ and ANR.

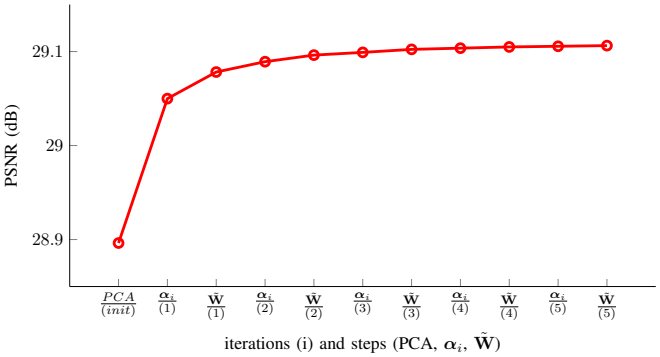


Fig. 2. Average PSNR [dB] on (Set14, $\times 3$) vs. number of iterations for learning our RB method, using 32 basis regressors and 16382 anchor points.

For the basis learning (Algorithm1), we fix the number of iterations to 4 because going further only slowly improves the performance (see Fig. 2). In Fig. 1 we compare the performance of RB when varying the number of regressors in the basis and the number of anchors. Note the logarithmic scale for the number of regressors. As expected the larger the number of regressors in the basis (plot from 16 to 128 basis regressors), the more accurate is RB in restoring HR details. The same goes with the number of anchors determining the partition granularity of the LR space. For clarity we plot the RB results only for 1024 and 16384 anchors, as the intermediate number of anchors (2048, 4096, 8192) provide intermediate results.

In Fig. 1 we also report the performance of A+ with the number of regressors varying from 16 to 16384 and for JOR [4] with 16, 32, and 64 regressors. Note that for A+ the number of regressors coincide with the number of anchors while for JOR the number of anchors is very large, fixed to the number of training samples, 5 millions. From the plot in Fig. 1, where the results for bicubic, Zeyde and SRCNN methods are provided for reference, we see that for equal number of regressors, our RB method is able to provide the best performance and when compared with JOR, if we look at the number of anchors, our RB uses hundreds of times fewer and is thus significantly more memory efficient. At the same time RB is closer to A+ in speed, while JOR is significantly slower.

D. Results

In Table I we report PSNR results for our RB as well as other state-of-the-art methods on the standard Set4, Set14 and B100 datasets for $\times 2$, $\times 3$ and $\times 4$ upscaling factors as commonly done in the literature [18], [6]. For our RB we use two settings with 32 and 64 regressors in the basis and 16384 anchor points. For A+ we report the results for the default 1024 anchor points and their corresponding 1024 anchored regressors, as well as, for 16384 anchors and corresponding

TABLE I

PSNR (DB) COMPARISON RESULTS ON STANDARD BENCHMARKS WITH UPSCALING FACTOR $\times 2$, $\times 3$, AND $\times 4$.

Dataset	scale	Bicubic	NE+LLE	Zeyde	ANR	SRCNN	JOR	A+ (1024 r.)	**RB (32 br.)**	**RB (64 br.)**	A+ (16384 r.)
Set5	x2	33.66	35.77	35.78	35.83	36.34	-	36.55	36.51	36.58	36.63
	x3	30.39	31.84	31.90	31.92	32.39	32.55	32.59	32.57	32.62	32.66
	x4	28.42	29.61	29.69	29.69	30.09	30.19	30.29	30.26	30.30	30.32
Set14	x2	30.23	31.76	31.81	31.80	32.18	-	32.28	32.22	32.26	32.30
	x3	27.54	28.60	28.67	28.65	29.00	29.09	29.13	29.11	29.14	29.17
	x4	26.00	26.81	26.88	26.85	27.20	27.26	27.32	27.29	27.32	27.35
B100	x2	29.56	30.77	30.78	30.82	31.14	-	31.18	31.16	31.19	31.20
	x3	27.21	27.93	27.97	27.97	28.21	28.27	28.29	28.28	28.31	28.32
	x4	25.96	26.50	26.55	26.54	26.71	26.79	26.82	26.80	26.82	26.84

TABLE II

MEMORY AND RUN-TIME COMPLEXITIES FOR ANCHORED REGRESSORS METHODS, WITHOUT A SEARCH STRUCTURE. N IS THE NUMBER OF ANCHORS, R IS THE NUMBER OF REGRESSORS, d_h IS THE HR PATCH SIZE, d_l IS THE LR PATCH FEATURES SIZE, P IS THE NUMBER OF PATCHES PROCESSED, AND o IS A PROCESSING OVERHEAD (*eg.* IMAGE UPSCALING, FILTERING, GRID AND FEATURES COMPUTATION) SHARED BY ALL THE METHODS.

Method	test complexity (number of basic operations)	memory (number of values)	default settings (for upscaling $\times 3$)
ANR [17]	$P(Nd_l + N + d_l d_h) + o$	$Nd_l + Rd_l d_h + 4d_l d_h$	$N = 1024$, $R = 1024$, $d_l = 30$, $d_h = 81$
A+ [18]	$P(Nd_l + N + d_l d_h) + o$	$Nd_l + Rd_l d_h + 4d_l d_h$	$N = 1024$, $R = 1024$, $d_l = 30$, $d_h = 81$
JOR [4]	$P(Nd_l + NlogN + Rk_{nn} + R + d_l d_h) + o$	$Nd_l + Rd_l d_h + 4d_l d_h + NR$	$N = 5000000$,$R = 32$,$d_l = 30$,$d_h = 81$,$k_{nn} = 16$
RB (ours)	$P(Nd_l + N + Rd_l d_h + d_l d_h) + o$	$Nd_l + Rd_l d_h + 4d_l d_h + NR$	$N = 1024$, $R = 32$, $d_l = 30$, $d_h = 81$

16384 anchored regressors. We use this setting to give a better understanding of the performance achieved by our RB method. RB with 32 regressors is close in PSNR performance to A+ with 1024, while with 64 regressors RB gets quite close to A+ with 16384 regressors and supports once more the previous reported results from Fig. 1. Besides A+, we compare with JOR in default settings (32 regressors and 5,000,000 anchors) and we note that in comparison our RB with the same number of regressors, 32, already achieves better performance, but as mentioned before, the number of anchors for our RB is much lower. Furthermore, we compare with other representative methods such as NE+LLE, Zeyde, ANR, SRCNN and bicubic interpolation, to whom we are generally superior in achieved PSNR performance.

For assessing the visual quality we show a couple of results on standard Set5 and Set14 images in Figs. 3, 4 and 5. We can see that the visual performance of our RB (32 basis regressors) is comparable with the compared A+ and slightly better than the other compared methods (JOR, SRCNN, ANR). The visual artifacts (like edge halos and blur) affect less A+ and our RB than the other methods, which is no surprise as RB builds upon A+ using the same principles but in a compact form.

E. Memory and run-time complexity

As shown in [17] for different neighbor embedding and sparse coding methods, most such methods are capable to reach a targeted performance level given that the proper parameters are employed. The difference is made by train and test time complexities and memory requirements. Therefore, we compare different SR methods not only directly through their PSNR performance but also by reporting their complexities at run-time (test) and memory requirements. In Table II we show the number of basic operations and the memory requirements for storing their models for our RB method and the most related anchored regression-based methods: ANR [17], A+ [18], and JOR [4]. A+ has the same complexity as ANR while improving the performance by learning the regressors from training data. At the same time, for the same number of anchors (N) and regressors (R) both JOR and RB require NR extra memory and both JOR and RB require extra operations *wrt.* to ANR/A+. However, usually JOR and RB use much fewer regressors than A+/ANR for comparable performance. For an equal number of anchors $N = 1024$, under the same conditions, if RB uses only 32 regressors instead of 1024 in A+ the memory footprint of RB is $\sim 16\times$ lower at the expense of requiring $3.2\times$ more operations at test. There is a trade-off between PSNR performance, memory savings, and runtime. On the other hand, if we compare A+ and RB, both with $N = 16,384$ anchors, they reach comparable average performance on Set14 and comparable runtime (RB requires 1.15 more operations than A+), but RB with only 32 regressors requires $36.5\times$ less memory than A+. When it comes to compare our RB to JOR, a method using a similar low number of stored regressors, we see that the huge number of anchors required by JOR ($N = 5,000,000$) makes RB faster at run-time, more than $281\times$ less memory demanding when using $N = 16384$ anchors and the same number $R = 32$ of regressors. All the methods can further benefit at run-time from using a search structure (*eg.* JOR uses a kd-tree structure and A+ uses a hierarchical search in [20]).

The performance of A+ significantly improves (0.3dB) with the increase (augmentation) of training data and with the increase of used regressors as shown in [20] for up to 65536 regressors learned from 50 million samples. This is a perfect setup for our method since our experiments shown that the relative memory reduction of RB increases with the increase number of regressors of A+ and the relative run-time overhead of RB diminishes.

V. CONCLUSION

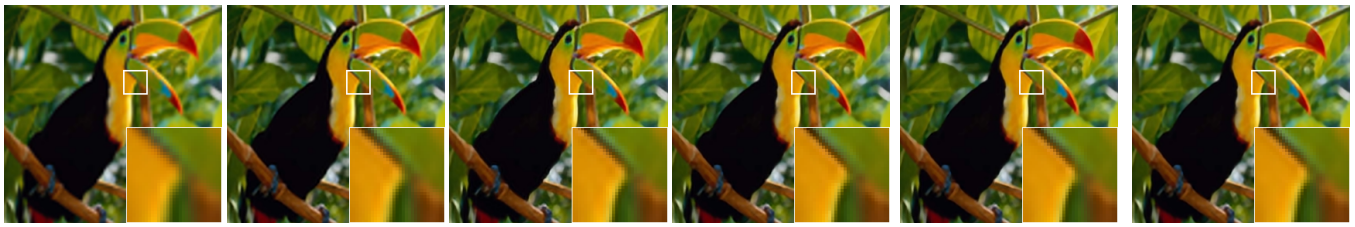
In this paper we proposed a novel Regressor Basis Learning (RB) method for anchored single-image super-resolution. RB jointly learns a compact regressor basis and a coding over the basis corresponding to a number of anchor points that

Bicubic / 33.91 dB ANR / 35.13 dB SRCNN / 35.01 dB JOR / 35.23 dB A+ (1024 r.) / 35.21 dB **RB (32 br.) / 35.30 dB**

Fig. 3. 'Baby' image from Set5 with upscaling $\times 3$. Best zoomed on screen.

Bicubic / 32.39 dB ANR / 33.82 dB SRCNN / 34.35 dB JOR / 34.65 dB A+ (1024 r.) / 34.73 dB **RB (32 br.) / 34.66 dB**

Fig. 4. 'Pepper' image from Set14 with upscaling $\times 3$. Best zoomed on screen.



Bicubic / 32.58 dB ANR / 34.60 dB SRCNN / 34.91 dB JOR / 35.54 dB A+ (1024 r.) / 35.54 dB **RB (32 br.) / 35.58 dB**

Fig. 5. 'Bird' image from Set5 with upscaling $\times 3$. Best zoomed on screen.

partition the low-resolution patch space. We are able to reach comparable performance with state-of-the-art methods such as A+ method whilst using a low memory footprint as the number of stored regressors for RB is order(s) of magnitude lower than the ones required by A+ for the same performance. This achievement makes our proposed method ideal for applications on memory constrained devices or under memory constrained scenarios.

ACKNOWLEDGMENT

This work was supported by the ERC project VarCity (#273940) and by the ETH General Fund (OK).

REFERENCES

[1] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. Alberi Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *BMVC*, 2012.

[2] H. Chang, D.-Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," *CVPR*, 2004.

[3] Z. Cui, H. Chang, S. Shan, B. Zhong, and X. Chen, "Deep network cascade for image super-resolution," in *ECCV*, 2014, pp. 49–64.

[4] D. Dai, R. Timofte, and L. Van Gool, "Jointly optimized regressors for image super-resolution," in *Eurographics*, 2015.

[5] D. Dai, Y. Wang, Y. Chen, and L. Van Gool, "Is image super-resolution helpful for other vision tasks?" in *WACV*, 2016.

[6] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *ECCV*, 2014.

[7] ——, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2015.

[8] W. Dong, L. Zhang, G. Shi, and X. Wu, "Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization," *TIP*, vol. 20, no. 7, pp. 1838–1857, 2011.

[9] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," *IEEE Computer Graphics and Applications*, vol. 22, no. 2, pp. 56–65, 2002.

[10] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *ICCV*, 2009.

[11] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *CVPR*, June 2015.

[12] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *CVPR*, 2016.

[13] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *ICCV*, 2001.

[14] E. Perez-Pellitero, J. Salvador, J. Ruiz-Hidalgo, and B. Rosenhahn, "Psyco: Manifold span reduction for super resolution," in *CVPR*, 2016.

[15] S. Schulter, C. Leistner, and H. Bischof, "Fast and accurate image upscaling with super-resolution forests," in *CVPR*, 2015, pp. 3791–3799.

[16] R. Timofte, "Anchored fusion for image restoration," in *23rd International Conference on Pattern Recognition (ICPR)*, 2016.

[17] R. Timofte, V. De Smet, and L. Van Gool, "Anchored neighborhood regression for fast example-based super resolution," in *ICCV*, 2013.

[18] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *Computer Vision–ACCV 2014*. Springer, 2014, pp. 111–126.

[19] R. Timofte, V. De Smet, and L. Van Gool, "Semantic super-resolution: When and where is it useful?" *Computer Vision and Image Understanding*, vol. 142, pp. 1 – 12, 2016.

[20] R. Timofte, R. Rothe, and L. Van Gool, "Seven ways to improve example-based single image super resolution," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[21] Z. Wang, D. Liu, J. Yang, W. Han, and T. Huang, "Deep networks for image super-resolution with sparse prior," in *ICCV*, 2015.

[22] C.-Y. Yang and M.-H. Yang, "Fast direct super-resolution by simple functions," in *ICCV*, 2013.

[23] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution as sparse representation of raw image patches," in *CVPR*, 2008.

[24] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Curves and Surfaces*, 2012, pp. 711–730.